

SDE-based Generative Models

Yuanzhi Zhu
SP in CVL

Content

- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

Content

- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

Noise Conditional Score Network (NCSN)*

Motivation: Learning the score function $s_\theta(x) \sim \nabla_x \log p(x)$ instead

Training Objective: Score Matching for Score Estimation

$$\frac{1}{2} \mathbb{E}_{p_{data}} [\| s_\theta(x) - \nabla_x \log p_{data}(x) \|_2^2] \longrightarrow \mathbb{E}_{p_{data}} \left[\underbrace{\text{tr}(\nabla_x s_\theta(x))}_{\text{expensive}} + \frac{1}{2} \| s_\theta(x) \|_2^2 \right]$$

Sampling with Langevin Dynamics

$$x_t = x_{t-1} + \frac{\epsilon}{2} \underbrace{\nabla_x \log p(x_{t-1})}_{\text{score}} + \sqrt{\epsilon} z_t, \quad \text{where } z_t \sim \mathcal{N}(0, \mathbf{I})$$

Noise Conditional Score Network (NCSN)

Cheaper Score Matching

1. Denoising Score Matching (DSM)

pre-specified noise distribution, i.e. Gaussian

$$\text{Matching perturbed data distribution } q_\sigma(\tilde{x}) = \int \underline{q_\sigma(\tilde{x}|x)p_{data}(x)} dx$$

$$\frac{1}{2} \mathbb{E}_{q_\sigma(\tilde{x}|x)p_{data}(x)} [\| s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) \|_2^2]$$

2. Sliced Score Matching: random projections to approximate $\text{tr}(\nabla_x s_\theta(x))$

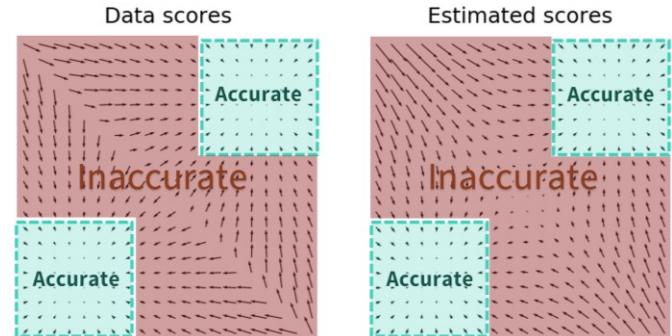
$$\mathbb{E}_{p_v} \mathbb{E}_{p_{data}} \left[v^T s_\theta(x)v - \frac{1}{2} \| s_\theta(x) \|_2^2 \right]$$

Noise Conditional Score Network (NCSN)

Low Density Regions Pitfalls

1. Inaccurate score estimation in low data density regions

For regions with $p_{data} \approx 0$, we do not have sufficient data samples for accurate estimation.

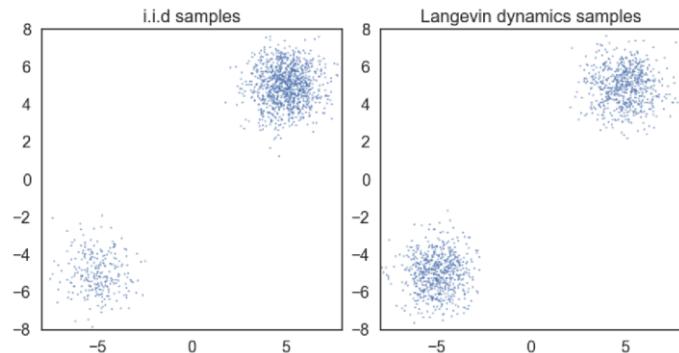


2. Slow mixing of Langevin dynamic

$$p_{data}(x) = \pi p_1(x) + (1 - \pi)p_2(x)$$

$$\nabla_x \log p_{data}(x) = \nabla_x \log p_1(x)$$

$$\nabla_x \log p_{data}(x) = \nabla_x \log p_2(x)$$



Noise Conditional Score Network (NCSN)

Training Objective: Score Matching a Sequence of Noise-levels

Large noise: perturbate the data sufficiently to better estimate the low density regions

Small noise: be able to converge to the true data distribution

Denoising score matching objective for given σ

$$q_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x} | x, \sigma^2 \mathbf{I}) \longrightarrow \nabla_x q_\sigma(\tilde{x}|x) = -(\tilde{x} - x)/\sigma^2$$

$$\ell(\theta; \sigma) \triangleq \frac{1}{2} \mathbb{E}_{p_{data}(x)} \mathbb{E}_{\tilde{x} \sim \mathcal{N}(x, \sigma^2 \mathbf{I})} \| s_\theta(\tilde{x}, \sigma) + \frac{\tilde{x} - x}{\sigma^2} \|_2^2$$

Final objective

$$\mathcal{L}(\theta; \{\sigma_i\}_{i=1}^L) \triangleq \frac{1}{L} \sum_{i=1}^L \underline{\lambda(\sigma_i)} \ell(\theta; \sigma_i)$$

coefficient function

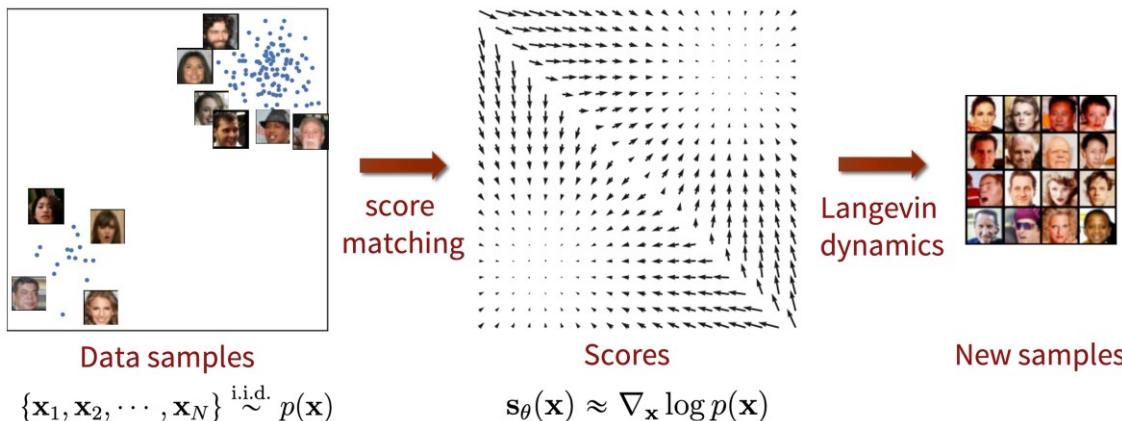
Noise Conditional Score Network (NCSN)

NCSN Inference: via Annealed Langevin Dynamics

A sequence of positive noise scales $\sigma_{min} = \sigma_1 < \sigma_2 < \dots < \sigma_L = \sigma_{max}$

$$p_{\sigma_{min}}(x) \approx p_{data}(x)$$

$$p_{\sigma_{max}}(x) \approx \mathcal{N}(x; 0, \sigma_{max}^2 \mathbf{I})$$



Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^L, \epsilon, T$.

- 1: Initialize $\tilde{\mathbf{x}}_0$
- 2: **for** $i \leftarrow L$ to 1 **do**
- 3: $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ $\triangleright \alpha_i$ is the step size.
- 4: **for** $t \leftarrow 1$ to T **do**
- 5: Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
- 6: $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_{\theta}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{z}_t$
- 7: **end for**
- 8: $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
- 9: **end for**
- return** $\tilde{\mathbf{x}}_T$

Content

- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

Deep Unsupervised Learning using Nonequilibrium Thermodynamics*

First Attempt from *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*

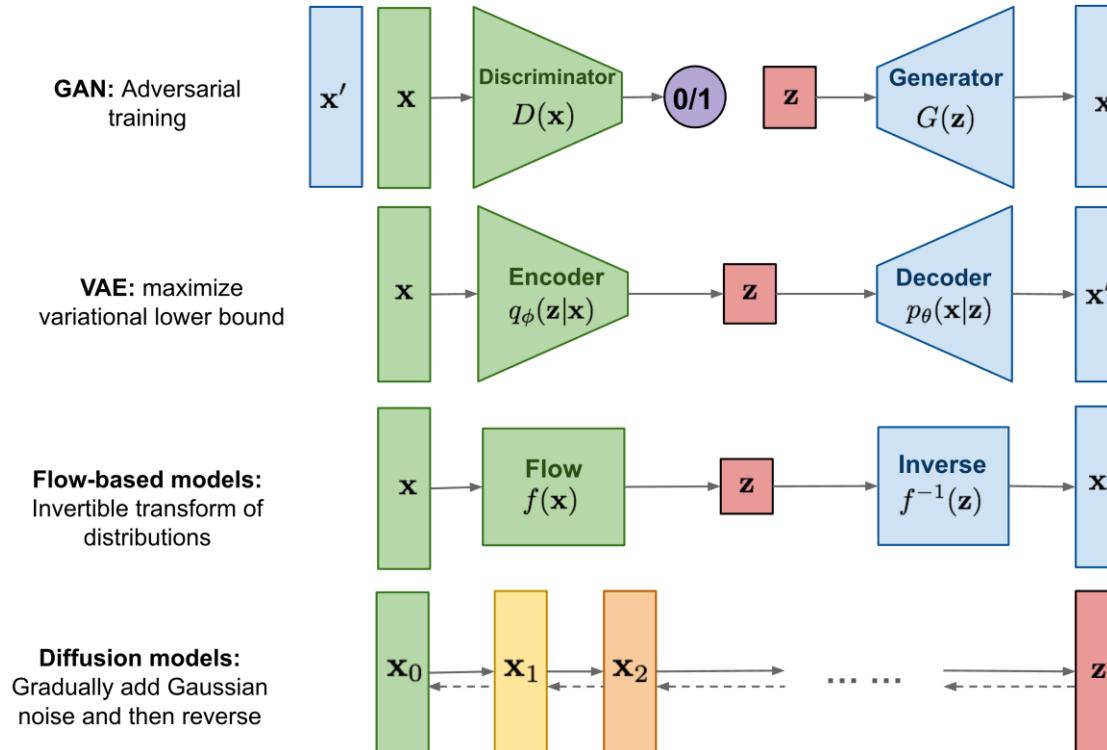
Main Ideas:

- inspired by non-equilibrium statistical physics
- systematically and slowly destroy structure in a data distribution (iterative forward diffusion)
- then **learn a reverse diffusion process** that restores structure in data(restore data distribution)

Math come soon in DDPM

Jascha Sohl-Dickstein is also author of *RealNVP* and *Score-based generative model through SDE*, now is working on ML theory and NLP

DDPM: Denoising Diffusion Probabilistic Models*



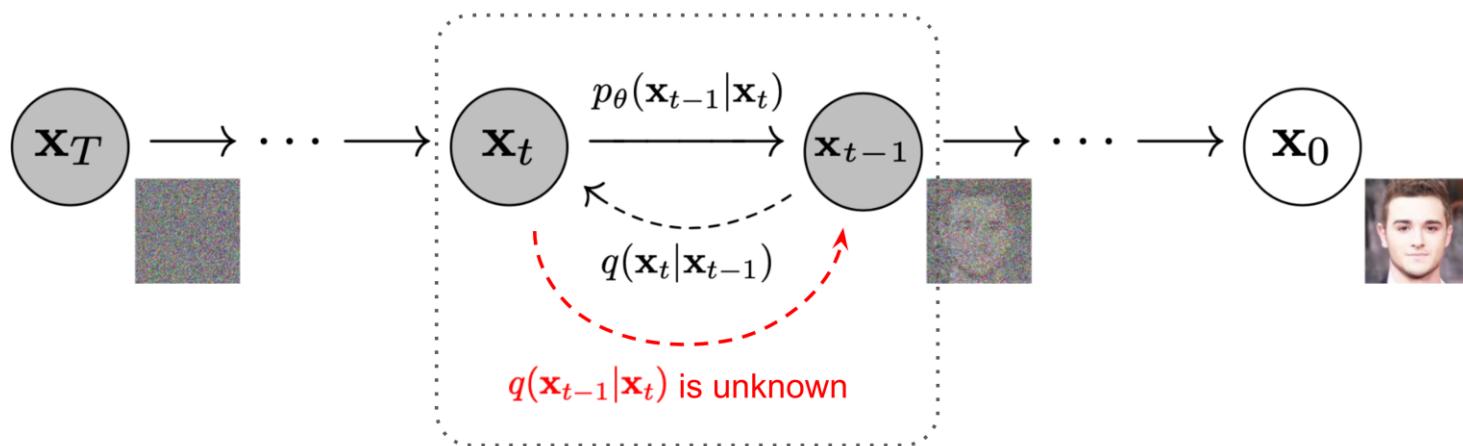
DDPM: Denoising Diffusion Probabilistic Models

True data dist. : $x_0 \sim q(x_0)$

Forward process: $q(x_{1:T}|x_0) := \prod_{t=1}^T q(x_t|x_{t-1})$ Markov Assumption

Reverse process: $p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)$

Use variational lower bound



DDPM: Denoising Diffusion Probabilistic Models

Forward Diffusion Process

$$q(x_{1:T}|x_0) := \prod_{t=1}^T q(x_t|x_{t-1})$$

Each Step

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \underbrace{\sqrt{1 - \beta_t}x_{t-1}}_{\text{norm invariant}}, \beta_t \mathbf{I}) \quad \text{or} \quad x_t = \sqrt{1 - \beta_t}x_{t-1} + \sqrt{\beta_t}z_{t-1}$$

variance schedule β_t controls the diffusion processing

For arbitrary t

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I}) \quad \alpha_t = 1 - \beta_t \text{ and } \bar{\alpha}_t = \prod_{i=1}^T \alpha_i$$

DDPM: Denoising Diffusion Probabilistic Models

Reverse Diffusion Process

if β_t is small enough, $q(x_{t-1}|x_t)$ will also be Gaussian

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t) \quad p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \boldsymbol{\Sigma}_\theta(x_t, t))$$

Reverse when condition on x_0

$$\begin{aligned} & q(x_{t-1}|x_t, x_0) \\ &= q(x_t|x_{t-1}, x_0) \frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} \\ &\propto \exp \left(-\frac{1}{2} \left(\frac{(x_t - \sqrt{\alpha_t} x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}} x_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\bar{\alpha}_t} x_0)^2}{1-\bar{\alpha}_t} \right) \right) \\ &= \exp \left(-\frac{1}{2} \left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}} \right) x_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t} x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} x_0 \right) x_{t-1} + C(x_t, x_0) \right) \right) \\ &\longrightarrow q(x_{t-1}|x_t, x_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t \mathbf{I}) \end{aligned}$$

$$\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \mathbf{z}_t \right)$$

$$\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

DDPM: Denoising Diffusion Probabilistic Models

Negative Log Likelihood to Variational Lower Bound

$$\begin{aligned} -\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\text{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\ &= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T} \sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)} \right] \\ &= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0) \right] \\ &= \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \\ \text{Let } L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \end{aligned}$$

DDPM: Denoising Diffusion Probabilistic Models

Parameterization for Training Loss

$$\begin{aligned}
 L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[\log \frac{q(\mathbf{x}_{1:T} | \mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] = \mathbb{E}_q \left[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} \right] \\
 &= \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_T | \mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \right] \xrightarrow{\text{Known}} \\
 &= \mathbb{E}_q \underbrace{[D_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) \| p_\theta(\mathbf{x}_T))]}_{L_T} + \sum_{t=2}^T \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_t} \underbrace{- \log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \\
 p_\theta(x_{t-1} | x_t) &= \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \boldsymbol{\Sigma}_\theta(x_t, t)) \quad \begin{cases} \boldsymbol{\mu}_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}z_\theta(x_t, t)) \\ \boldsymbol{\Sigma}_\theta(x_t, t) = \sigma_t^2 \mathbf{I} \end{cases}
 \end{aligned}$$

Model The Noise(Residual)

$$L_t^{\text{simple}} = \mathbb{E}_{x_0, z_t} \left[\|z_t - z_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z_t, t)\|^2 \right]$$

DDPM: Denoising Diffusion Probabilistic Models

Nonetheless, it is just **another parameterization** of $p_\theta(x_{t-1}|x_t)$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \boldsymbol{\Sigma}_\theta(x_t, t)) \quad \begin{cases} \boldsymbol{\mu}_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}z_\theta(x_t, t)) \\ \boldsymbol{\Sigma}_\theta(x_t, t) = \sigma_t^2 \mathbf{I} \end{cases}$$

Algorithm 1 Training

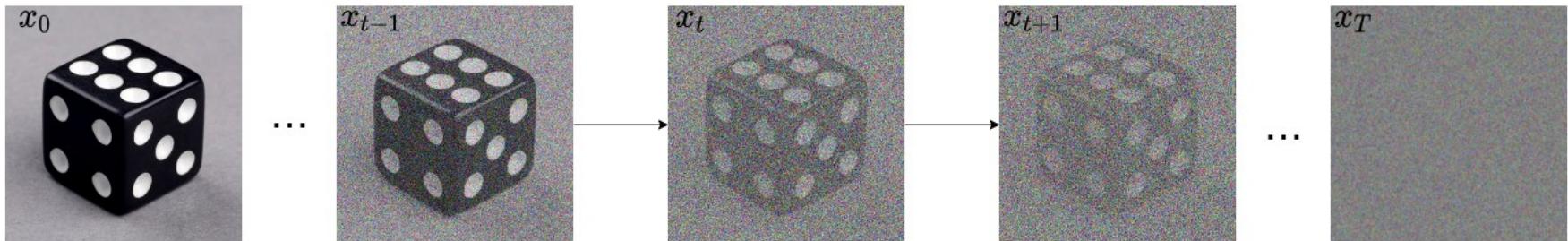
```
1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
       $\nabla_\theta \|\boldsymbol{\epsilon} - \mathbf{z}_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2$ 
6: until converged
```

Algorithm 2 Sampling

```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \mathbf{z}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

DDPM: Denoising Diffusion Probabilistic Models

$$x_0 \sim q(x_0) \quad \longrightarrow \quad q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I) \quad \longrightarrow \quad q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1})$$



$$p_\theta(x_0) = \int p(x_T) \prod_{i=1}^T p_\theta(x_{t-1}|x_t) dx_{1:T} \quad \longleftarrow \quad p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad \longleftarrow \quad x_T \sim \mathcal{N}(0, I)$$

Content

- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

DDIM: Denoising Diffusion Implicit Models*

Variational Inference for *Non-Markovian* Forward Processes

$$-\log p_\theta(\mathbf{x}_0) \leq -\log p_\theta(\mathbf{x}_0) + D_{\text{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0) \| p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0))$$

Model this directly

$$q_\sigma(x_{1:T}|x_0) = q_\sigma(x_T|x_0) \prod_{t=2}^T q_\sigma(x_{t-1}|x_t, x_0)$$

Reverse Process: deterministic given x_t, x_0

$$x_{t-1} = \sqrt{\alpha_{t-1}}x_0 + \sqrt{1 - \alpha_{t-1}} \cdot \epsilon_{t-1} = \sqrt{\alpha_{t-1}}x_0 + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \epsilon_t + \sigma_t \epsilon$$

$$q_\sigma(x_{t-1}|x_t, x_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}x_0 + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \frac{x_t - \sqrt{\alpha_t}x_0}{\sqrt{1 - \alpha_t}}, \sigma_t^2 \mathbf{I}\right)$$

Forward: still Gaussian (non-Markovian)

$$q_\sigma(x_t|x_{t-1}, x_0) = \frac{q_\sigma(x_{t-1}|x_t, x_0)q_\sigma(x_t|x_0)}{q_\sigma(x_{t-1}|x_0)}$$

DDIM: Denoising Diffusion Implicit Models

Given: noisy observation x_t

Prediction of the corresponding $\hat{x}_0(x_t) = f_\theta^{(t)}(x_t) = (x_t - \sqrt{1 - \alpha_t} \epsilon_\theta^{(t)}(x_t)) / \sqrt{\alpha_t}$

Model difference between x_0 and x_t

$$p_\theta^{(t)}(x_{t-1}|x_t) = \begin{cases} \mathcal{N}(f_\theta^{(1)}(x_1), \sigma_1^2 I) & \text{if } t = 1 \\ q_\sigma(x_{t-1}|x_t, f_\theta^{(t)}(x_t)) & \text{otherwise} \end{cases}$$

Variational Inference Objective (equivalent to objective in DDPM for certain weights)

$$\begin{aligned} J_\sigma(\epsilon_\theta) &:= \mathbb{E}_{\mathbf{x}_{0:T} \sim q_\sigma(\mathbf{x}_{0:T})} [\log q_\sigma(\mathbf{x}_{1:T}|\mathbf{x}_0) - \log p_\theta(\mathbf{x}_{0:T})] \\ &= \mathbb{E}_{\mathbf{x}_{0:T} \sim q_\sigma(\mathbf{x}_{0:T})} \left[\log q_\sigma(\mathbf{x}_T|\mathbf{x}_0) + \sum_{t=2}^T \log q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) - \sum_{t=1}^T \log p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t) - \log p_\theta(\mathbf{x}_T) \right] \end{aligned}$$

Surrogate Objective $L_t^{\text{simple}} = \mathbb{E}_{x_0, z_t} \left[\|z_t - z_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} z_t, t)\|^2 \right]$

DDIM: Denoising Diffusion Implicit Models

Sampling from Generalized Generative Processes $p_\theta^{(t)}(x_{t-1}|x_t)$

$$x_{t-1} = \underbrace{\sqrt{\alpha_{t-1}} \left(\frac{x_t - \sqrt{1 - \alpha_t} \epsilon_\theta^{(t)}(x_t)}{\sqrt{\alpha_t}} \right)}_{\text{"predicted } x_0\text{ "}} + \underbrace{\sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \epsilon_\theta^{(t)}(x_t)}_{\text{"direction pointing to } x_t\text{ "}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$$

$$\sigma_t := \eta \sqrt{(1 - \alpha_{t-1}) / (1 - \alpha_t)} \sqrt{1 - \alpha_t / \alpha_{t-1}}$$

- **DDPM:** $\eta = 1$ (forward process becomes Markovian (different noise schedule from [vanilla DDPM](#)))
- **DDIM:** $\eta = 0$ (forward process becomes deterministic)

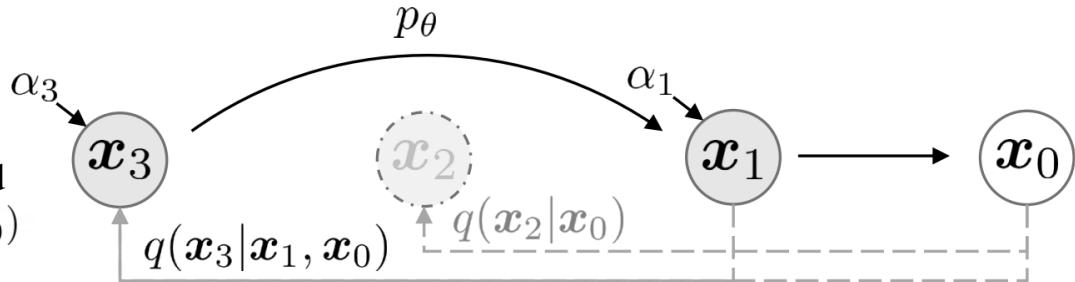
Table 1: CIFAR10 and CelebA image generation measured in FID. $\eta = 1.0$ and $\hat{\sigma}$ are cases of **DDPM** (although [Ho et al. \(2020\)](#) only considered $T = 1000$ steps, and $S < T$ can be seen as simulating DDPMs trained with S steps), and $\eta = 0.0$ indicates **DDIM**.

S	CIFAR10 (32×32)					CelebA (64×64)				
	10	20	50	100	1000	10	20	50	100	1000
0.0	13.36	6.84	4.67	4.16	4.04	17.33	13.73	9.17	6.53	3.51
0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
$\hat{\sigma}$	367.43	133.37	32.72	9.99	3.17	299.71	183.83	71.71	45.20	3.26

DDIM: Denoising Diffusion Implicit Models

Accelerated Generation Processes

Denoising surrogate objective does not depend on the specific forward procedure $q_\sigma(x_{t-1}|x_0)$



Consider the forward process as defined on a subset $\tau = [\tau_1, \tau_2, \dots, \tau_{\dim(\tau)}] \subset [1, 2, \dots, T]$

$$q_{\sigma, \tau}(x_{\tau_{i-1}}|x_{\tau_t}, x_0) = \mathcal{N}\left(x_{\tau_{i-1}}; \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \frac{x_{\tau_i} - \sqrt{\bar{\alpha}_t}x_0}{\sqrt{1 - \bar{\alpha}_t}}, \sigma_t^2 \mathbf{I}\right)$$

The generative process now samples latent variables according to $\text{reversed}(\tau)$, which we term (sampling) *trajectory*

→ Train a model with arbitrary number forward steps but only sample from some of them in the generative process

DDIM: Denoising Diffusion Implicit Models

Relevance to Neural ODEs

$$x_{t-1} = \underbrace{\sqrt{\alpha_{t-1}} \left(\frac{x_t - \sqrt{1-\alpha_t} \epsilon_\theta^{(t)}(x_t)}{\sqrt{\alpha_t}} \right)}_{\text{"predicted } x_0 \text{ "}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(x_t)}_{\text{"direction pointing to } x_t \text{ "}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$$

$$\implies \frac{x_{t-\Delta t}}{\sqrt{\alpha_{t-\Delta t}}} - \frac{x_t}{\sqrt{\alpha_t}} = \left(\sqrt{\frac{1-\alpha_{t-\Delta t}}{\alpha_{t-\Delta t}}} - \sqrt{\frac{1-\alpha_t}{\alpha_t}} \right) \epsilon_\theta^{(t)}(x_t)$$

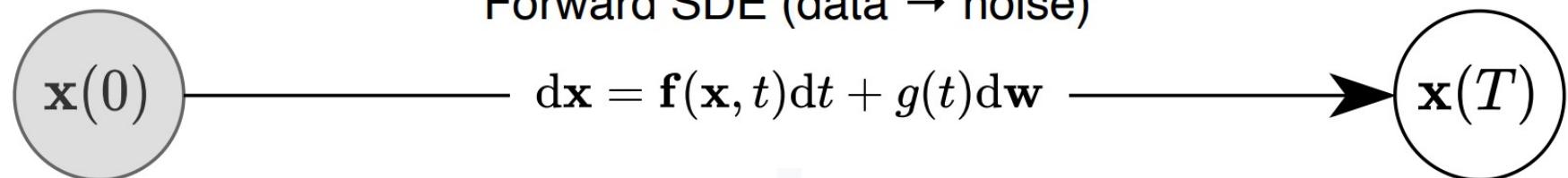
$$\begin{aligned} \sigma &:= \sqrt{\frac{1-\alpha}{\alpha}} & \implies d\bar{x}(t) &= \epsilon_\theta^{(t)} \left(\frac{\bar{x}(t)}{\sqrt{\sigma^2 + 1}} \right) d\sigma(t) & \text{Variance-Exploding SDE soon} \\ \bar{x} &:= \frac{x}{\sqrt{\alpha}} \end{aligned}$$

Content

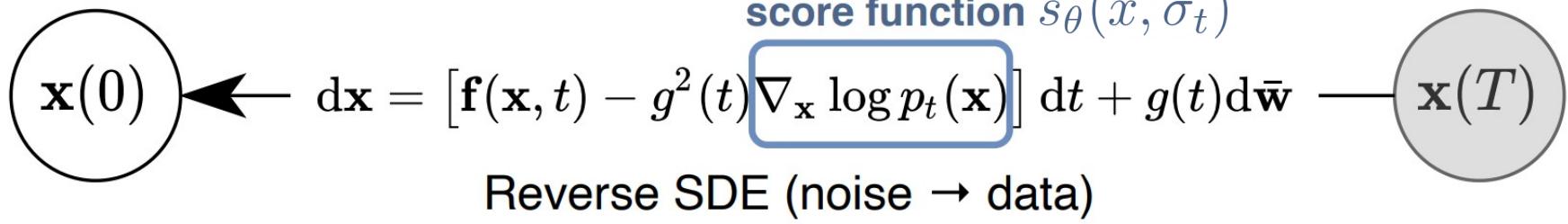
- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

SDE-based Generative Models: A Unified Framework*

Forward SDE (data → noise)



score function $s_\theta(x, \sigma_t)$



Reverse SDE (noise → data)

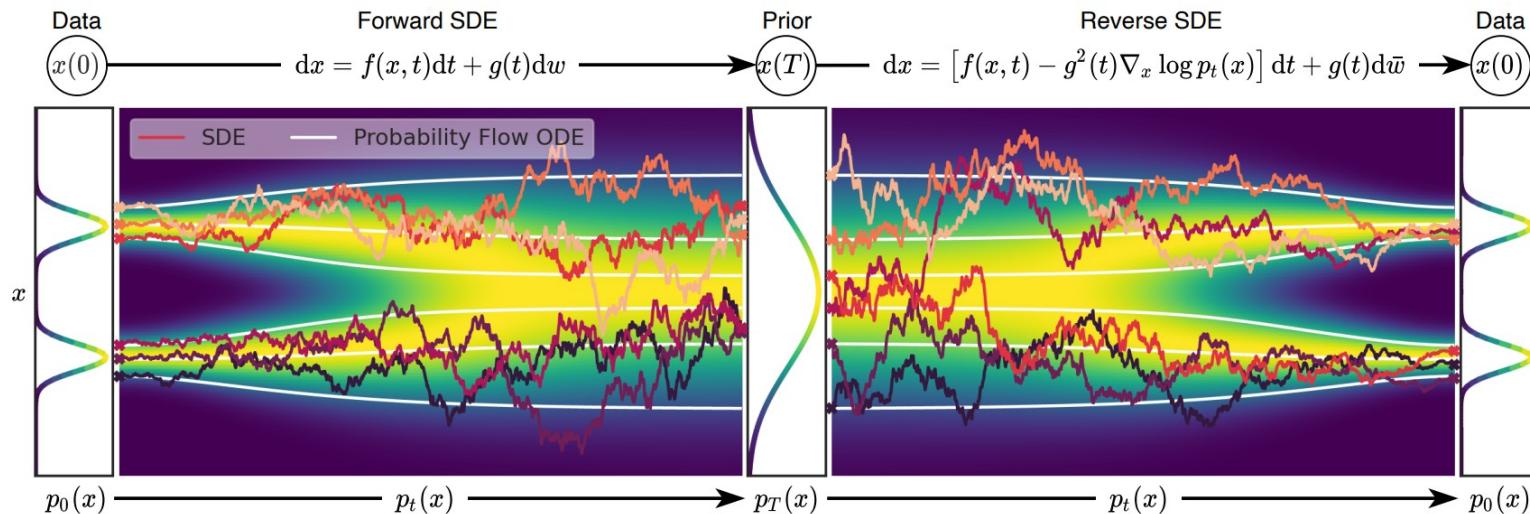
SDE-based Generative Models: A Unified Framework

Model diffusion process as solution of Itô SDE (continuous time)

$$dx = f(x, t)dt + g(t)dw$$

Generating samples by reversing the SDE

$$dx = [f(x, t) - g(t)^2 \nabla_x \log p_t(x)]dt + g(t)d\bar{w}$$



SDE-based Generative Models: A Unified Framework

Training Objective (DSM)

$$\theta^* = \arg \min \mathbb{E}_{t \sim U(0, T)} \left\{ \lambda(t) \mathbb{E}_{x(0)} \mathbb{E}_{x(t)|x(0)} [\| s_\theta(x(t), t) - \nabla_{x(t)} \log p_{0t}(x(t)|x(0)) \|_2^2] \right\}$$

known Gaussian when $f(x, t)$ if affine

Discretizations

$$dx = f(x, t)dt + g(t)dw$$

SDE Form	Discrete Markov Chain	SDE Expression
Variance Exploding (VE) SDE (NCSN)	$x_i = x_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} z_{i-1}$	$dx = \sqrt{\frac{d[\sigma^2(t)]}{dt}} dw$
Variance Preserving (VP) SDE (DDPM)	$x_i = \sqrt{1 - \beta_i} x_{i-1} + \sqrt{\beta_i} z_{i-1}$	$dx = \frac{1}{2} \beta(t) x dt + \sqrt{\beta(t)} dw$

SDE-based Generative Models: A Unified Framework

Reverse SDE Discretization $x_i = x_{i+1} - f_{i+1}(x_{i+1}) + g_{i+1}g_{i+1}^T s_\theta(x_{i+1}, i+1) + g_{i+1}z_{i+1}$

Predictor-Corrector (PC) Samplers

1. **Predictor:** general-purpose numerical SDE solvers
2. **Corrector:** score-based MCMC (i.e. Langevin MCMC)

DDPM: predictor only
NCSN: corrector only

Algorithm 2 PC sampling (VE SDE)

```
1:  $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ 
2: for  $i = N - 1$  to 0 do
3:    $\mathbf{x}'_i \leftarrow \mathbf{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2) \mathbf{s}_{\theta} * (\mathbf{x}_{i+1}, \sigma_{i+1})$ 
4:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:    $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2} \mathbf{z}$ 
6:   for  $j = 1$  to  $M$  do
7:      $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
8:      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta} * (\mathbf{x}_i, \sigma_i) + \sqrt{2\epsilon_i} \mathbf{z}$ 
9:   return  $\mathbf{x}_0$ 
```

Algorithm 3 PC sampling (VP SDE)

```
1:  $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $i = N - 1$  to 0 do
3:    $\mathbf{x}'_i \leftarrow (2 - \sqrt{1 - \beta_{i+1}}) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta} * (\mathbf{x}_{i+1}, i+1)$ 
4:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:    $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}} \mathbf{z}$ 
6:   for  $j = 1$  to  $M$  do
7:      $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
8:      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta} * (\mathbf{x}_i, i) + \sqrt{2\epsilon_i} \mathbf{z}$ 
9:   return  $\mathbf{x}_0$ 
```

Predictor

Corrector

SDE-based Generative Models: A Unified Framework

Relationship between **Bayesian Posterior** and **Reverse SDE**

$$\mathbf{x}_{t+\Delta t} - \mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_t)\Delta t + g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (2)$$

$$p(\mathbf{x}_{t+\Delta t} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+\Delta t}; \mathbf{x}_t + \mathbf{f}_t(\mathbf{x}_t)\Delta t, g_t^2 \Delta t \mathbf{I}) \\ \propto \exp\left(-\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - \mathbf{f}_t(\mathbf{x}_t)\Delta t\|^2}{2g_t^2 \Delta t}\right) \quad (3)$$

$$p(\mathbf{x}_t | \mathbf{x}_{t+\Delta t}) = \frac{p(\mathbf{x}_{t+\Delta t} | \mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{x}_{t+\Delta t})} = p(\mathbf{x}_{t+\Delta t} | \mathbf{x}_t) \exp(\log p(\mathbf{x}_t) - \log p(\mathbf{x}_{t+\Delta t})) \\ \propto \exp\left(-\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - \mathbf{f}_t(\mathbf{x}_t)\Delta t\|^2}{2g_t^2 \Delta t} + \log p(\mathbf{x}_t) - \log p(\mathbf{x}_{t+\Delta t})\right) \quad (4)$$

$$\log p(\mathbf{x}_{t+\Delta t}) \approx \log p(\mathbf{x}_t) + (\mathbf{x}_{t+\Delta t} - \mathbf{x}_t) \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \Delta t \frac{\partial}{\partial t} \log p(\mathbf{x}_t) \quad (5)$$

$$p(\mathbf{x}_t | \mathbf{x}_{t+\Delta t}) \propto \exp\left(-\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - [\mathbf{f}_t(\mathbf{x}_t) - g_t^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)]\Delta t\|^2}{2g_t^2 \Delta t}\right) \\ \approx \exp\left(-\frac{\|\mathbf{x}_t - \mathbf{x}_{t+\Delta t} + [\mathbf{f}_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - g_{t+\Delta t}^2 \nabla_{\mathbf{x}_{t+\Delta t}} \log p(\mathbf{x}_{t+\Delta t})]\Delta t\|^2}{2g_{t+\Delta t}^2 \Delta t}\right) \quad (7)$$

$$d\mathbf{x} = [\mathbf{f}_t(\mathbf{x}) - g_t^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g_t d\mathbf{w} \quad (8)$$

SDE-based Generative Models: A Unified Framework

Sampling: DDPM and SDE point of views (equivalent up to first order)

The ancestral sampling of DDPM matches its reverse diffusion counterpart when $\beta_i \approx 0$ for all i

Bayesian Posterior

Reverse SDE

$$\begin{aligned}\mathbf{x}_i &= \frac{1}{\sqrt{1 - \beta_{i+1}}} (\mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1)) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\ &= \left(1 + \frac{1}{2} \beta_{i+1} + o(\beta_{i+1})\right) (\mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1)) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\ &\approx \left(1 + \frac{1}{2} \beta_{i+1}\right) (\mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1)) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\ &= \left(1 + \frac{1}{2} \beta_{i+1}\right) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1) + \frac{1}{2} \beta_{i+1}^2 \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\ &\approx \left(1 + \frac{1}{2} \beta_{i+1}\right) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\ &= \left[2 - \left(1 - \frac{1}{2} \beta_{i+1}\right)\right] \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\ &\approx \left[2 - \left(1 - \frac{1}{2} \beta_{i+1}\right) + o(\beta_{i+1})\right] \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1} \\ &= (2 - \sqrt{1 - \beta_{i+1}}) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}} \mathbf{z}_{i+1}.\end{aligned}$$

SDE-based Generative Models: A Unified Framework

Model: DDPM and SDE point of views

Score in score-based model is affine transformation of predicted noise in DDPM

$$\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon} \quad \text{Equivalent one step forward}$$

$$\begin{aligned} s_\theta(\mathbf{x}_t, t) &\approx \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t | \mathbf{x}_0) && \text{Denoising score matching} \\ &= -\frac{\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0}{\bar{\beta}_t^2} && \text{Gaussian assumption} \\ &= -\frac{\boldsymbol{\varepsilon}}{\bar{\beta}_t} \\ &\approx -\frac{\boldsymbol{\varepsilon}_t(\mathbf{x}_t, t)}{\bar{\beta}_t} \end{aligned}$$

SDE-based Generative Models: A Unified Framework

ODE form

Fokker-Plank function associated with forward diffusion

$$\frac{\partial}{\partial t} p_t(\mathbf{x}) = -\nabla_{\mathbf{x}} \cdot [\mathbf{f}_t(\mathbf{x}) p_t(\mathbf{x})] + \frac{1}{2} g_t^2 \nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}} p_t(\mathbf{x})$$

With $\sigma_t^2 \leq g_t^2$ and $\begin{aligned} \mathbf{f}_t &\longrightarrow \mathbf{f}_t(\mathbf{x}) - \frac{1}{2}(g_t^2 - \sigma_t^2) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \\ g_t &\longrightarrow \sigma_t \end{aligned}$

$$d\mathbf{x} = \mathbf{f}_t(\mathbf{x}) dt + g_t d\mathbf{w} \quad \xleftarrow{\text{equivalent}} \quad d\mathbf{x} = \left(\mathbf{f}_t(\mathbf{x}) - \frac{1}{2}(g_t^2 - \sigma_t^2) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right) dt + \sigma_t d\mathbf{w}$$

Reverse $d\mathbf{x} = \left(\mathbf{f}_t(\mathbf{x}) - \frac{1}{2}(g_t^2 + \sigma_t^2) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right) dt + \sigma_t d\mathbf{w}$

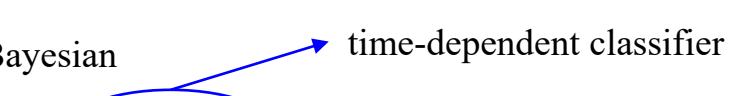
ODE $d\mathbf{x} = \left(\mathbf{f}_t(\mathbf{x}) - \frac{1}{2} g_t^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right) dt$

Comparing with SDEs, ODEs can be solved with larger step sizes as they have no randomness.

SDE-based Generative Models: A Unified Framework

Controllable Generation

$$dx = [f(x, t) - g(t)^2 \nabla_x \log p_t(x|y)]dt + g(t)d\bar{w}$$

Bayesian  time-dependent classifier 

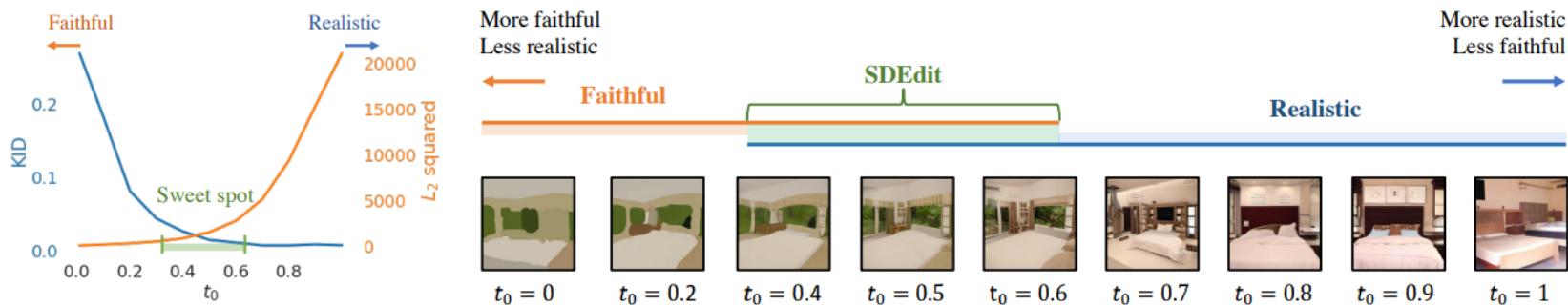
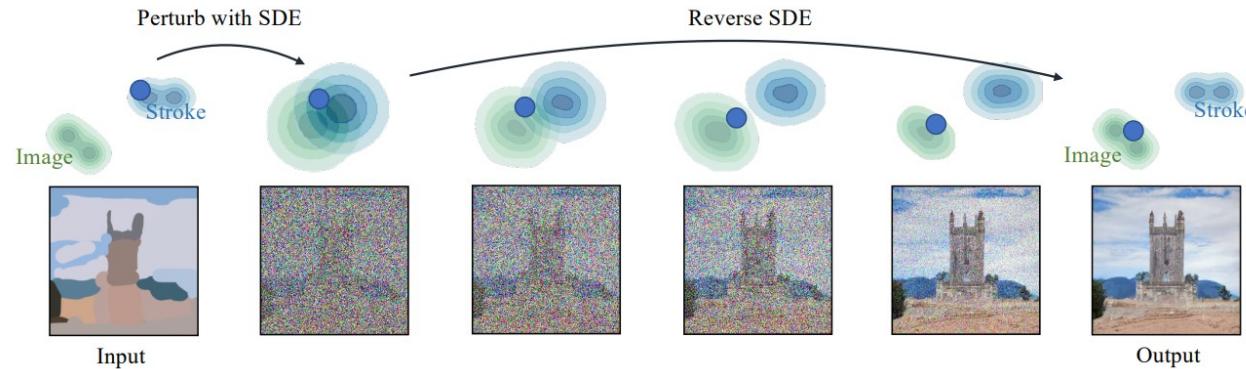
$$dx = \{f(x, t) - g(t)^2 [\nabla_x \log p_t(x) + \nabla_x \log p_t(y|x)]\}dt + g(t)d\bar{w}$$



Content

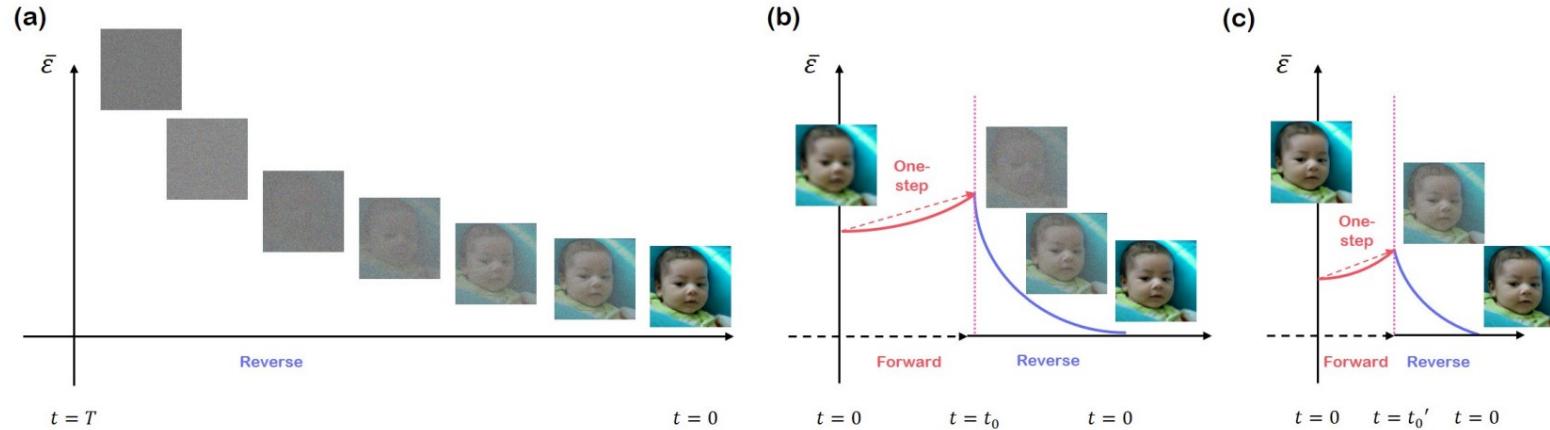
- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

SDEdit: Guided Image Synthesis and Editing with SDE*



Come-Closer-Diffuse-Faster: Accelerating Conditional Diffusion Models for Inverse Problems through Stochastic Contraction*

Same idea but different downstream tasks: super-resolution (SR), inpainting, and MRI reconstruction



Accelerating Diffusion Models via Early Stop of the Diffusion Process*

Get not fully noising image by diffusing output from pre-trained models like GAN and VAE

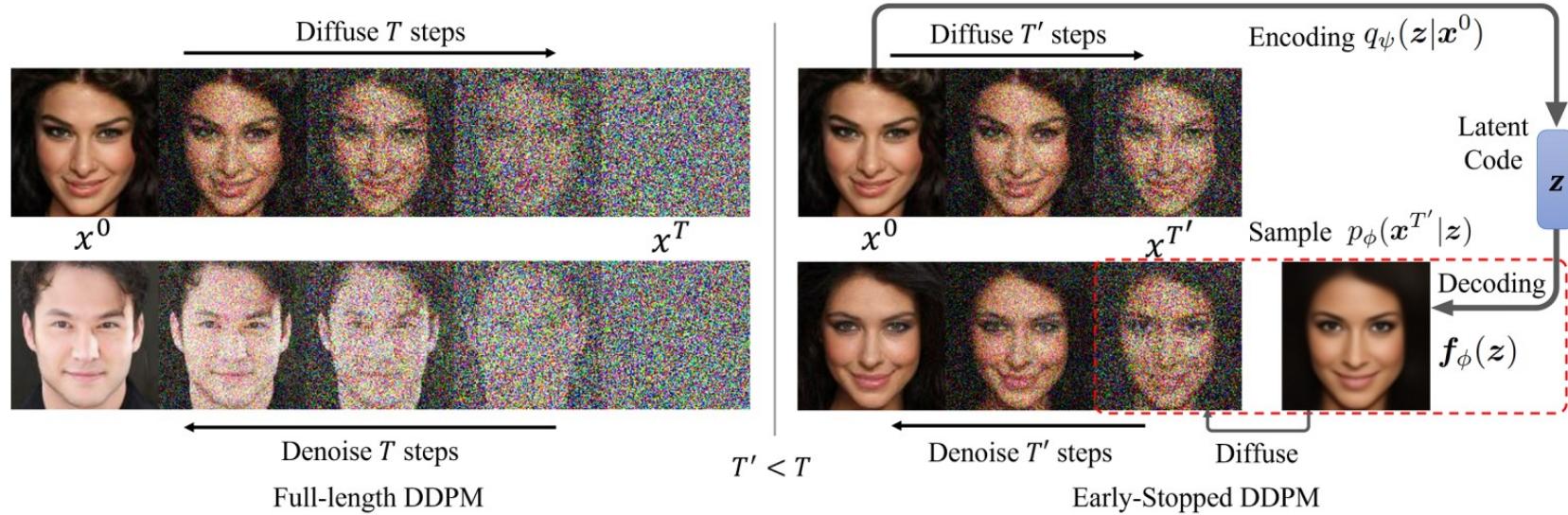
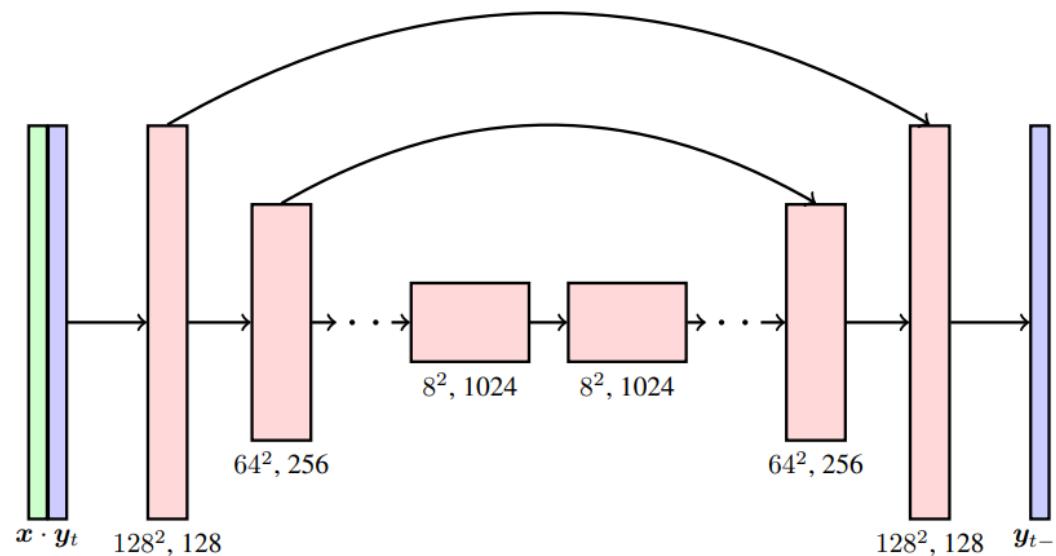


Image Super-Resolution via Iterative Refinement*

The condition is concatenated with y_t along the channel dimension (cascaded)



Same author also proposed palette for multi-tasks[†], same architecture used for cascaded diffusion[‡]

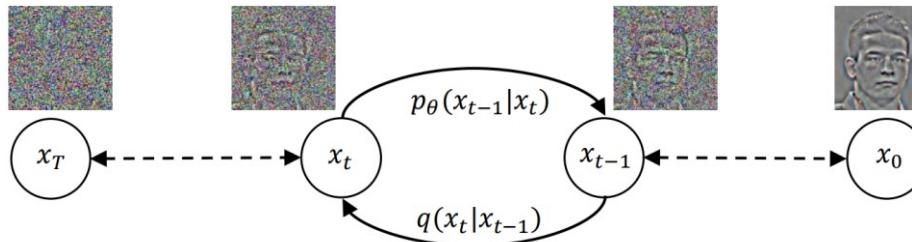
*[2104.07636] Image Super-Resolution via Iterative Refinement (arxiv.org/)

†[2111.05826] Palette: Image-to-Image Diffusion Models (arxiv.org/)

‡[2106.15282] Cascaded Diffusion Models for High Fidelity Image Generation (arxiv.org/)

SRDiff: Single Image Super-Resolution with Diffusion Probabilistic Models*

Learn the residual with condition encoded LR (fused as 2D CNN block outputs)



Algorithm 1 Training

- 1: **Input:** LR image and its corresponding HR image pairs $P = \{(x_L^k, x_H^k)\}_{k=1}^K$, total diffusion step T
- 2: **Initialize:** randomly initialized conditional noise predictor ϵ_θ and pretrained LR encoder \mathcal{D}
- 3: **repeat**
- 4: Sample $(x_L, x_H) \sim P$
- 5: Upsample x_L as $up(x_L)$, compute $x_r = x_H - up(x_L)$
- 6: Encode LR image x_L as $x_e = \mathcal{D}(x_L)$
- 7: Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $t \sim \text{Uniform}(\{1, \dots, T\})$
- 8: Take gradient step on

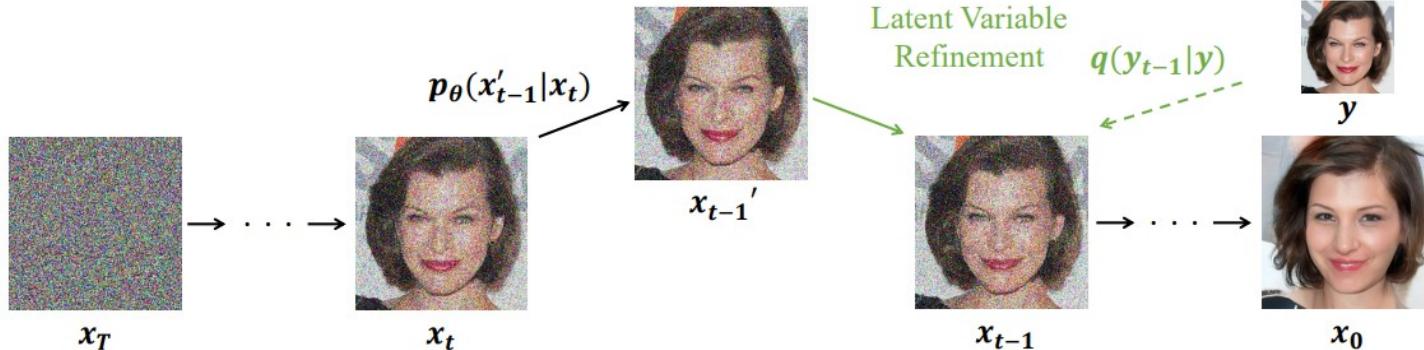
$$\nabla_\theta \|\epsilon - \epsilon_\theta(x_t, x_e, t)\|, x_t = \sqrt{\bar{\alpha}_t} x_r + \sqrt{1 - \bar{\alpha}_t} \epsilon$$
- 9: **until** converged

Algorithm 2 Inference

- 1: **Input:** LR image x_L , total diffusion step T
- 2: **Load:** conditional noise predictor ϵ_θ and LR encoder \mathcal{D}
- 3: Sample $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 4: Upsample x_L to $up(x_L)$
- 5: Encode LR image x_L as $x_e = \mathcal{D}(x_L)$
- 6: **for** $t = T, T-1, \dots, 1$ **do**
- 7: Sample $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $z = 0$
- 8: Compute x_{t-1} using Eq. (7):

$$x_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, x_e, t) \right) + \sigma_\theta(x_t, t) z$$
- 9: **end for**
- 10: **return** $x_0 + up(x_L)$ as SR prediction

ILVR: Conditioning Method for DDPM*

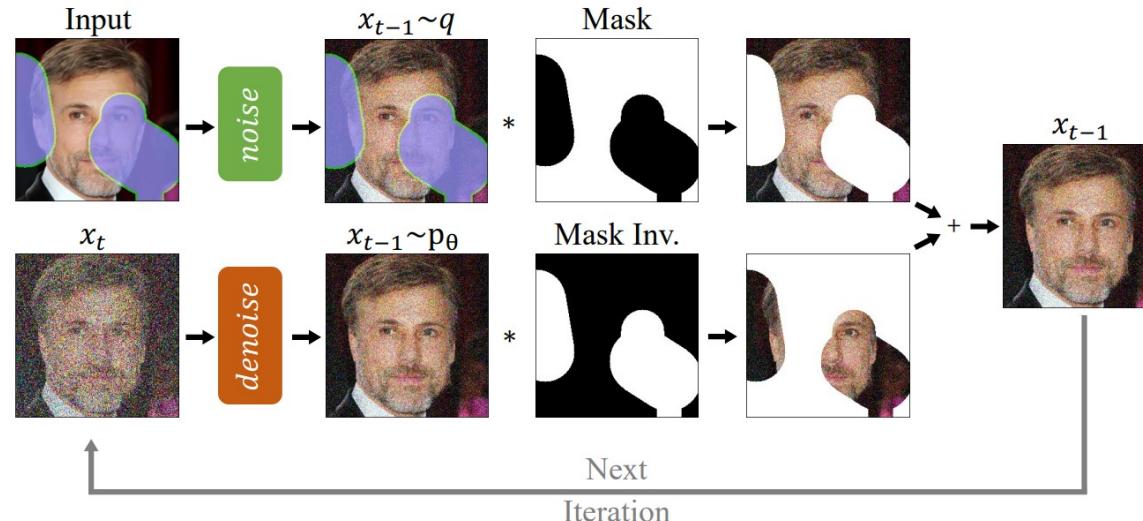


Algorithm 1 Iterative Latent Variable Refinement

```
1: Input: Reference image  $y$ 
2: Output: Generated image  $x$ 
3:  $\phi_N(\cdot)$ : low-pass filter with scale N
4: Sample  $x_T \sim N(\mathbf{0}, \mathbf{I})$ 
5: for  $t = T, \dots, 1$  do
6:    $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$ 
7:    $x'_{t-1} \sim p_\theta(x'_{t-1}|x_t)$        $\triangleright$  unconditional proposal
8:    $y_{t-1} \sim q(y_{t-1}|y)$              $\triangleright$  condition encoding
9:    $x_{t-1} \leftarrow \phi_N(y_{t-1}) + x'_{t-1} - \phi_N(x'_{t-1})$ 
10: end for
11: return  $x_0$ 
```

RePaint: Inpainting using Denoising Diffusion Probabilistic Models*

Same idea but different downstream tasks from ILVR



Algorithm 1 Inpainting using our RePaint approach.

```
1:  $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:   for  $u = 1, \dots, U$  do
4:      $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\epsilon = \mathbf{0}$ 
5:      $x_{t-1}^{\text{known}} = \sqrt{\bar{\alpha}_t} x_0 + (1 - \bar{\alpha}_t) \epsilon$ 
6:      $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $z = \mathbf{0}$ 
7:      $x_{t-1}^{\text{unknown}} = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z$ 
8:      $x_{t-1} = m \odot x_{t-1}^{\text{known}} + (1 - m) \odot x_{t-1}^{\text{unknown}}$ 
9:     if  $u < U$  and  $t > 1$  then
10:       $x_t \sim \mathcal{N}(\sqrt{1 - \beta_{t-1}} x_{t-1}, \beta_{t-1} \mathbf{I})$ 
11:    end if
12:  end for
13: end for
14: return  $x_0$ 
```

Noise Estimation for Generative Diffusion Models*

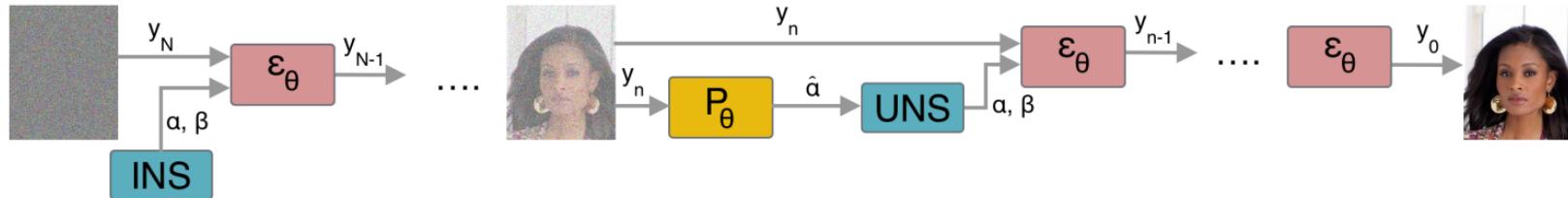


Figure 1: An overview of our generative process. INS and UNS are respectively the functions initializeNoiseSchedule() and updateNoiseSchedule($\hat{\alpha}$).

Algorithm 3: P_θ training procedure

```

1: repeat
2:    $y_0 \sim q(y_0)$ 
3:    $s \sim \mathcal{U}(\{1, \dots, N\})$ 
4:    $\sqrt{\bar{\alpha}} \sim \mathcal{U}([l_{s-1}, l_s])$ 
5:    $\varepsilon \sim \mathcal{N}(0, I)$ 
6:    $y_s = \sqrt{\bar{\alpha}}y_0 + \sqrt{1 - |\bar{\alpha}|}\varepsilon$ 
7:    $\hat{\alpha} = P_\theta(y_s)$ 
8:   Take gradient descent step on:  

     $\| \log(1 - \bar{\alpha}) - \log(1 - \hat{\alpha}) \|_2$ 
9: until converged

```

Algorithm 4: Model inference procedure

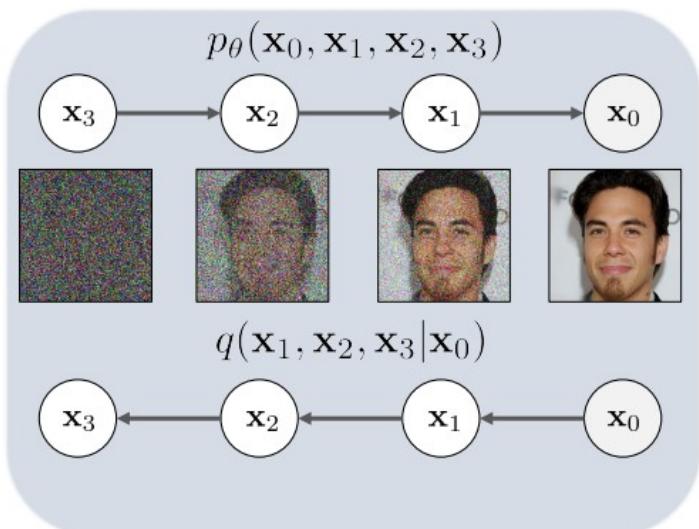
```

1:  $N$  Number of iterations
2:  $y_N \sim \mathcal{N}(0, I)$ 
3:  $\alpha, \beta = \text{initialNoiseSchedule}()$ 
4: for  $n = N, \dots, 1$  do
5:    $z \sim \mathcal{N}(0, I)$ 
6:    $\hat{\varepsilon} = \varepsilon_\theta(y_n, \sqrt{\bar{\alpha}_n})$  or  $\varepsilon_\theta(y_n, t)$  where  $\bar{\alpha}_n \in [l_t, l_{t-1}]$ 
7:    $y_{n-1} = \frac{y_n - \frac{1 - \alpha_n}{\sqrt{1 - \bar{\alpha}_n}}\hat{\varepsilon}}{\sqrt{\alpha_n}}$ 
8:   if  $n \in U$  then
9:      $\hat{\alpha} = P_\theta(y_{n-1})$ 
10:     $\alpha, \beta, \tau = \text{updateNoiseSchedule}(\hat{\alpha}, n)$ 
11:   end if
12:   if  $n \neq 1$  then
13:      $y_{n-1} = y_{n-1} + \sigma_n z$ 
14:   end if
15: end for
16: return  $y_0$ 

```

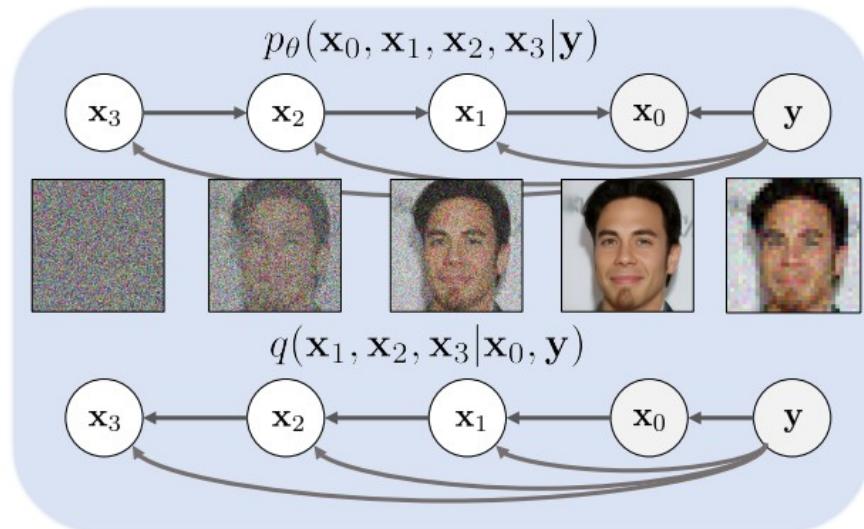
Denoising Diffusion Restoration Models (DDRM)*

An efficient, unsupervised posterior sampling method



Denoising Diffusion Probabilistic Models
(Independent of inverse problem)

Use pre-trained
models for linear
inverse problems



Denoising Diffusion Restoration Models
(Dependent on inverse problem)

Dual Diffusion Implicit Bridges for Image-to-Image Translation*

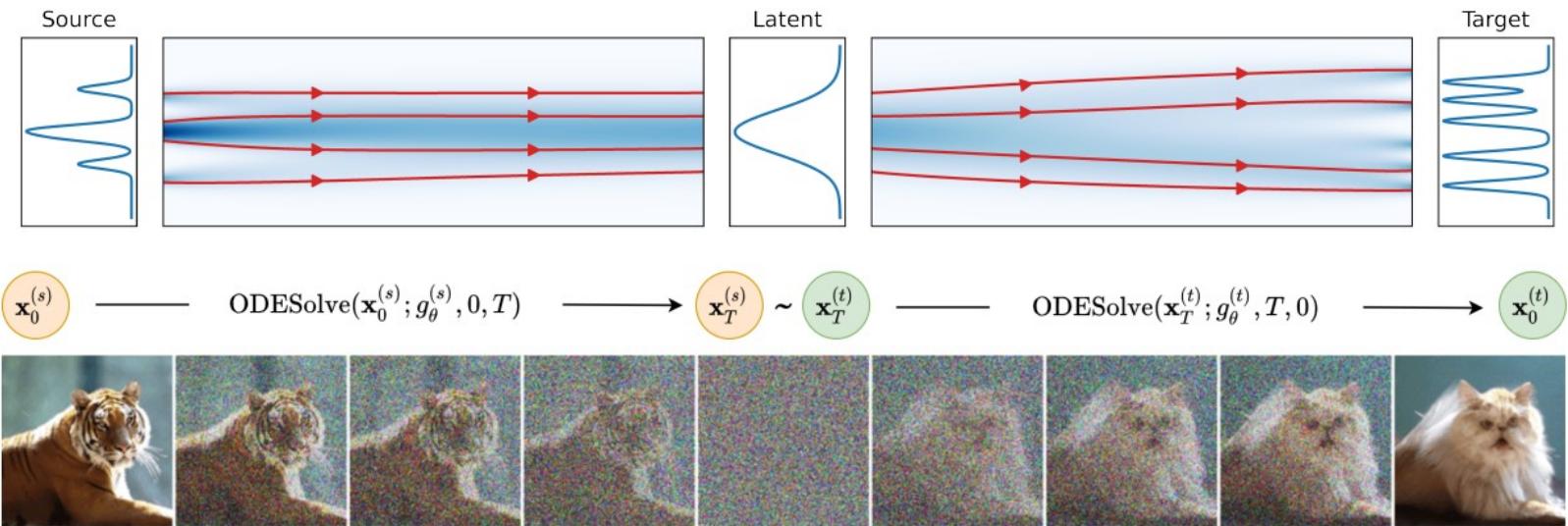
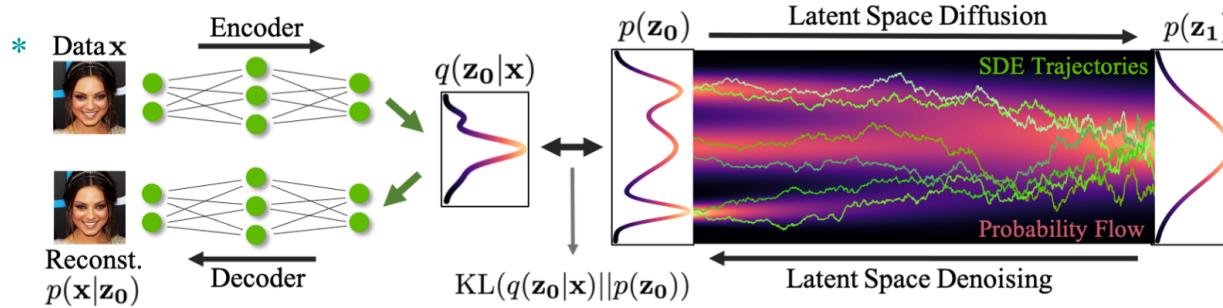
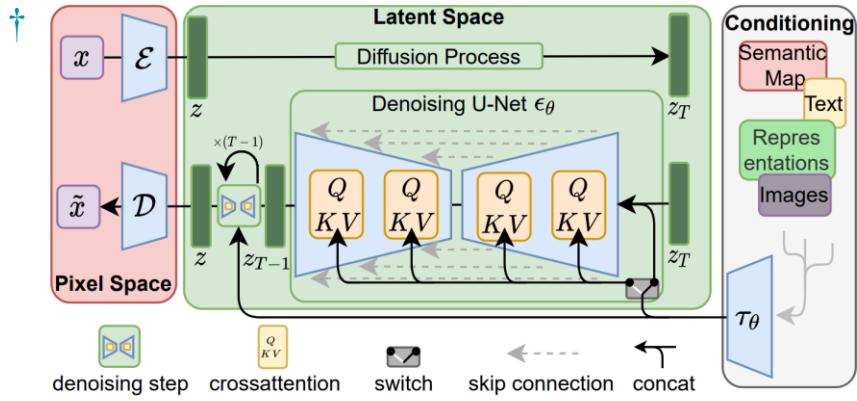


Figure 1: **Dual Diffusion Implicit Bridges:** DDIBs leverage two ODEs for image translation. Given a source image $\mathbf{x}_0^{(s)}$, the source ODE runs in the forward direction to convert it to the latent $\mathbf{x}_T^{(s)}$, while the target, reverse ODE then constructs the target image $\mathbf{x}_0^{(t)}$. (Top) Illustration of the DDIB idea between two one-dimensional distributions. (Bottom) DDIB from a tiger to a cat using a pre-trained conditional diffusion model.

Score-based Generative Modeling in Latent Space*

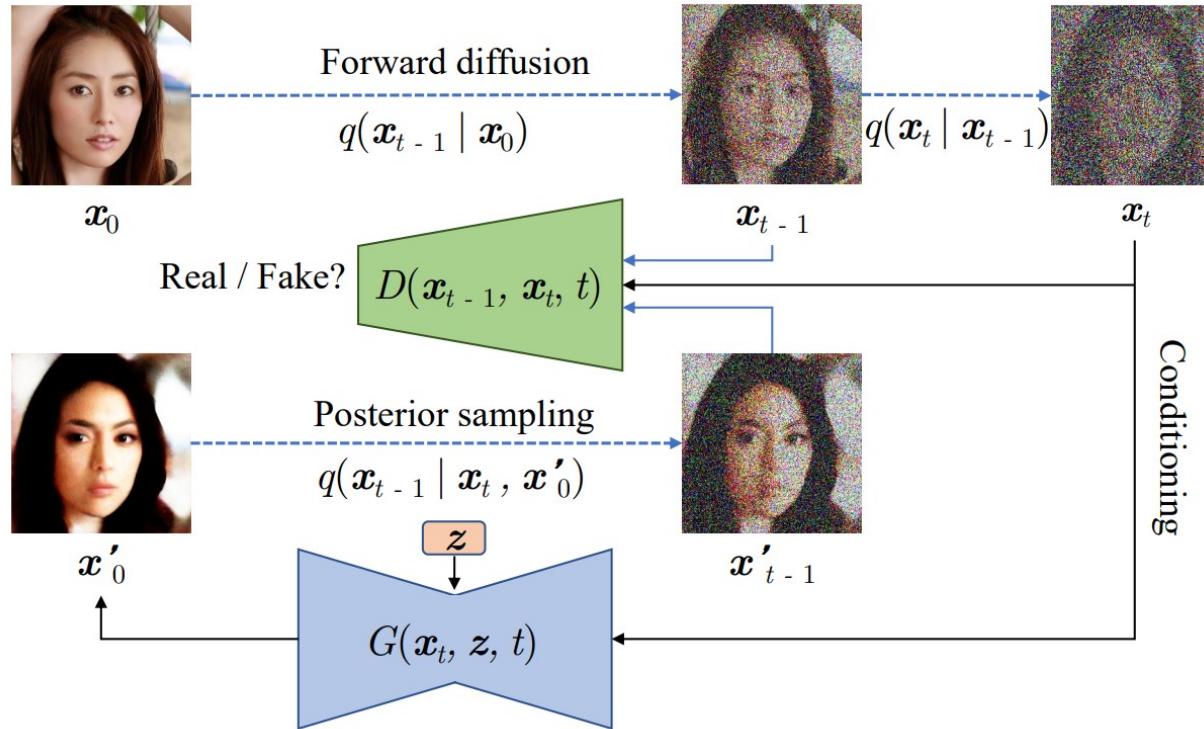
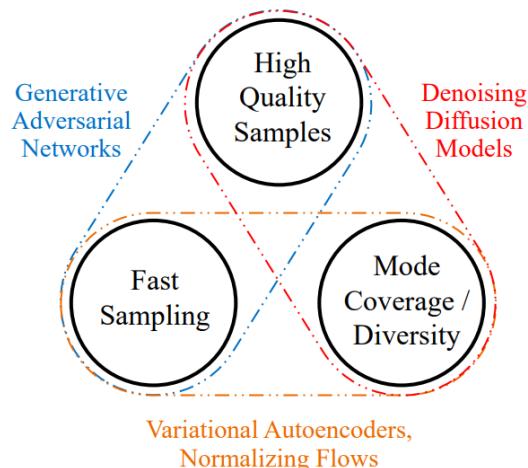
Faster diffusion
in latent space



†[2112.10752] High-Resolution Image Synthesis with Latent Diffusion Models (arxiv.org)

*[2106.05931] Score-based Generative Modeling in Latent Space (arxiv.org)

Tackling the Generative Learning Trilemma with Denoising Diffusion GANS*



Thank You!

Useful Resources

- [What's the score? – Review of latest Score Based Generative Modeling papers](#)
- [zhangbaijin/Diffusion-model-low-level \(github.com\)](#)
- [What are Diffusion Models? | Lil'Log \(lilianweng.github.io\)](#)
- [Generative Modeling by Estimating Gradients of the Data Distribution | Yang Song](#)
- [Diffusion Models as a kind of VAE](#)
- [yang-song/score_sde_pytorch](#)
- [Denoising Diffusion Probabilistic Models \(DDPM\) \(labml.ai\)](#)
- [Denoising Diffusion-based Generative Modeling: Foundations and Applications](#)